Jogo 2048

Implementação Assembly

Rodrigo Dallagnol, Thiago M. de Melo

Universidade de Caxias do Sul (UCS) – Caxias do Sul – RS – Brasil

{rdallagnol1,tmmelo}@ucs.br

1. Problema

O objetivo desse trabalho consiste no desenvolvimento do jogo 2048 na linguagem de programação Assembly. O jogo consiste em uma matriz quadrada com 16 posições que, no início do jogo, apresenta todas casas vazias por exceção de uma onde se um contra um número, dois ou quatro. O usuário consegue mover os blocos nas quatro direções principais, cima, baixo, esquerda e direita. A cada movimento realizado duas coisas importantes acontecem: os blocos se deslocam e um novo número surge.

Quanto a movimentação, todos os blocos se movem sentido indicado pelo jogador, mas sem sobrepor os outros valores, a menos que dois números iguais colidam se agrupando de forma que o novo bloco surge possuindo a soma de seus valores. Após o final da movimentação, um novo número surge em uma posição determina de forma pseudoaleatória em uma das posições livres, o valor deste bloco é determinado de forma pseudoaleatória sendo os valores iniciais possíveis dois ou quatro. Na Fig. 1 é possível visualizar um exemplo de movimentação, em que na esquerda há a tela original e a direita têm-se o resultado após a movimentação para a direção esquerda bem como o novo número gerado. Caso após o surgimento de um novo bloco o usuário não possua nenhum movimento possível, ou seja, todas as posições da matriz estão ocupadas e não há números iguais em posições adjacentes, o jogador perde o jogo.

Por outro lado, o jogador vence o jogo se formar o bloco contendo o número 2048. Além disso é computado o número de jogadas e um escore para cada jogo, em que esse escore é incrementado toda vez que há a junção de blocos, sendo o valor de incremento correspondente ao valor do bloco resultante da junção.

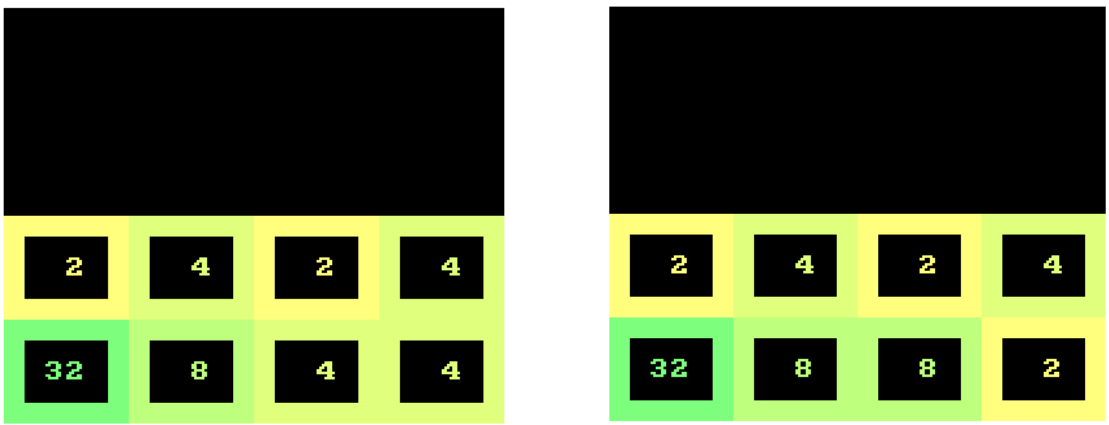

Figura 1. Exemplo de movimentação e matriz de jogo.

1.1 Requisitos

Além dos requisitos básicos que envolvem o funcionamento do jogo descrito acima, têm-se algumas exigências. O jogo deve possuir uma tela inicial contendo as opções: Jogar 2048, Recordes, Automático 2048 e Sair, onde a seleção é feita pressionando a primeira letra de cada opção.

Ao selecionar a opção Jogar 2048 o jogo deve se iniciar. Na tela do jogo há, na lateral, o escore do jogador no jogo atual, o número de jogadas realizas até então e o maior escore já obtido. Centralizado na tela deve haver uma matriz 4x4 em que se dá o jogo. Cada bloco deve possuir uma cor única, sendo que os blocos começam em tons de amarelo e vão até tons de azul, passando pelo verde, conforme o valor do bloco aumenta.

A tela de recordes deve apresentar o nome, escore e número de jogadas dos jogadores que obtiveram as cinco maiores pontuações já registradas. O limite para o nome é de 10 caracteres.

Ao escolher o modo automático, o usuário deve ser redirecionado para uma página na qual pode informar o número de vezes *n* que deseja que a máquina jogue de forma automática. O programa deve armazenar o menor número de jogadas necessárias para alcançar cada expoente maior ou igual a cinco, ou seja, valor de bloco 32. Observa-se que o menor valor é obtido comparando o número de jogadas necessárias em cada jogo dos *n* jogos determinado pelo usuário. Uma vez que a máquina terminou de jogar todos os *n* jogos, deve aparecer uma tela contendo o número de jogadas necessárias para alcançar aquele valor de bloco, bem como o escore no momento do surgimento do bloco daquele expoente.

Ao finalizar o jogo, caso o jogador tenha obtido pontuação capaz de entrar nos recordes, o usuário deve ser redirecionado para uma tela onde poderá informar seu nome, de modo que este será devidamente posicionado entres os cinco melhores.

2. Implementação

Nesta seção iremos discutir os principais aspectos da implementação do programa. Realizando uma explicação geral do algoritmo, das heurísticas do modo automático, uma lista de funções (*procs*) existentes no código, explicando as principais e por fim uma lista dos blocos me memória reservados.

2.1. Algoritmo

Logo após a execução do arquivo do jogo, antes de exibir o menu inicial, o jogo realiza o carregamento dos blocos de cada valor de potência de dois, começando em dois e indo até 2048. Os blocos, por exceção do bloco todo preto que corresponde a célula vazia, são criados através de um laço e transferindo da memória de vídeo para um bloco de memória reservado para esta finalidade. Uma vez feito isso, inicializa-se o menu, conforme mostra Fig 2.

Figura 2. Tela do Menu Inicial

Na opção Jogar 2048 a tela inicial, que pode ser visualizada na Fig 3, é carregada, e na sequencias os seguintes passos são realizado de forma cíclica: aguarda o pressionamento de uma tecla, verifica se é uma tecla válida, ou seja direcional para mover ou *esc* para sair, chama a *proc* de tratamento para aquela tecla que, para as direcionais, realiza a movimentação de todos os blocos no sentido indicado agrupando os blocos conforme a regra, quando há movimentação ou a mescla de blocos é acionada uma *flag* de movimento, realizada a impressão da nova matriz na tela e o incremento no número de jogadas, sendo que, em caso de mesclas, há ainda a chamada da função que incrementa o escore. Caso a *flag* de movimento estiver ativa, um novo número é gerado, as *flags* são limpas e há a checagem de fim de jogo, caso ainda seja possível realizar movimento começa-se o ciclo novamente, caso contrário vai para tela de final de jogo.

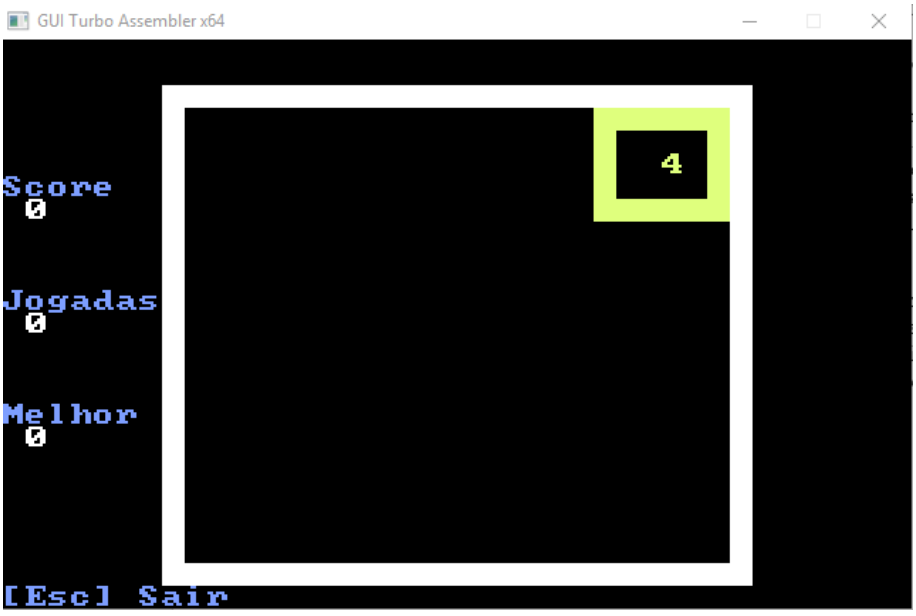

Figura 3. Tela inicial do jogo.

Para garantir que um número formado naquela jogada não seja mesclado com outro bloco na mesma rodada foi utilizada uma matriz secundária, em que um significa que aquela posição corresponde a um bloco formado na jogada atual e zero um bloco que já existia antes e, portanto, passível de ser mesclado. A cada rodada essa matriz é zerada.

2.2. Modo Automático

A principal ideia do modo automático é manter o maior número na posição inferior direita, para isto a máquina fica realizando movimentos alternados para direita e baixo, sendo que quando não é possível mover para essas direções move-se para esquerda, caso não seja possível mover para esquerda move-se para cima, caso não seja possível mover para cima significa que o jogo terminou.

Para realizar o controle do número de jogadas necessárias para formar cada bloco contendo números acima de 32, após cada jogada é verificado se há a presença do expoente procurado, inicialmente 5, caso encontre o expoente procurado se verifica se o número de jogadas é menor que o número de jogadas armazenado como o melhor até então para aquele expoente, realizando a substituição se for o caso. Uma vez que o expoente é encontrado o valor de busca é incrementado de modo que agora se procura pelo próximo expoente.

2.3. Lista de Procs

O programa é constituído de um total de 31 procs, sendo elas: stop, readchar, delay, backspace_clear, clear_memory, read_number, write_number, clear_screen, create_square, set_memory, print_game_header, print_mat, show_menu, getkey_menu, show_hiscore, reset_game, start_bot, bot_mode, manage_bot, start_game, update_header, new_number, getkey_game, add_score, arrow_up, arrow_down, arrow_left, arrow_right, check_gameover, update_hiscore, clear_flags. Descreveremos na sequência o funcionamento das principais.

Delay – Utiliza a interrupção 15h gerar um intervalo de tempo, o qual será utilizado para dar um tempo entre as movimentações dos blocos.

Create_Square – Essa proc imprime um retângulo de cinco linhas e seis colunas da cor contida em *bh* que possui canto superior esquerdo na linha zero, coluna zero. Na sequência um quadrado preto é impresso com três linhas e quatro colunas cujo canto superior esquerdo localiza-se na linha um, coluna um. Para realizar a impressão destes quadrados é utilizada a interrupção 10h. Em seguida o número referente ao bloco é impresso em seu centro utilizando-se da mesma interrupção. Por fim armazena na memória a configuração de pixels existentes na área do bloco na memória de vídeo.

Set_Memory – Proc que configure a memória necessária para o jogo, isto é, todos os blocos. Sendo que o bloco referente ao expoente zero é todo preto e criado de maneira a parte, e os demais blocos foram criados utilizando-se de um laço com a proc Create_Square.

Print_Mat – É responsável por imprimir a matriz do jogo, para isto ela tem que localizar em memória os pixels referentes ao bloco de cada expoente presente na matriz e transferi-lo para memória de vídeo. Aqui utiliza-se os blocos de memória squares e

position_config, em que estão armazenados, respectivamente, os pixels de todos os blocos e a posição referente a cada célula da matriz. Sabe-se que cada bloco ocupa 1920 bytes então se desejamos imprimir, por exemplo, o bloco correspondente ao expoente 3, basta multiplica 1920 por 3 e somar ao offset de squares.

Getkey_Menu – Realiza a captura da tecla pressionada enquanto na tela de menu, para isto utiliza-se a interrupção 16h.

Bot_Mode – Principal função do modo automático, aqui está contido a lógica que a máquina utilizará, chamando as funções que realizam os movimentos dos blocos conforme a direção desejada.

Manage_Bot – Realiza o armazenamento do menor número de jogadas para alcançar cada expoente acima de cinco durante a repetição do modo automático no número de vezes determinada pelo usuário.

Update_Header – Atualiza o escore, número de jogadas durante o jogo. Caso o seu escore atual supere o melhor já registrado, o melhor escore passa a ser atualizado concomitantemente.

New_Number – Responsável por gerar números pseudoaleatórios baseados no relógio (*clock*) do processador, captado através da interrupção 1Ah. Estes números servem para posicionar o novo número gerado bem como determinar se ele é dois ou quatro.

Getkey_Game – Análoga ao getkey_menu, realiza a captura de tecla pressionada durante o jogo.

Arrow_Up, Arrow_Down, Arrow_Left, Arrow_Right – Todas funcionam de maneira similar. Realizam a movimentação dos blocos na direção indiciada pelo usuário, bem como a mescla dos blocos.

Check_GameOver – Verifica, após o surgimento de cada novo número, se é possível continuar jogando, checando se todas as posições estão ocupadas, e em caso positivo verifica se existem dois números iguais adjacentes.

2.4. Blocos de Memória

Matriz db 16 dup(0) – Armazena os expoentes presentes em cada posição. É a matriz do jogo.

Join_flag db 16 dup(0) – Matriz auxiliar utilizada para controlar quais blocos já foram unidos naquela jogada, impedindo que ocorra duas uniões envolvendo o mesmo bloco na mesma rodada.

Mov_flag db 0 – Acionada quando houve a movimentação ou união de algum bloco a cada tecla pressionada.

Score dw 0 – Escore do jogador no jogo atual.

Jogadas dw 0 – Número de jogadas realizadas no jogo atual.

Melhor dw 0 – Contém o melhor desempenho obtido por algum jogador.

Best_players db 50 dup(32) – Armazena os nomes dos jogadores nos recordes.

Best_scores dw 5 dup (0) – Armazena os escore dos jogos nos recordes.

Best_jogadas dw 5 dup (0) – Armazena o número de jogadas referente aos jogos presentes nos recordes.

Squares db 23040 dup(0) – Armazena configuração de pixels de todos os blocos utilizados no jogo. Cada bloco corresponde a 1920 bytes.

Numbers_config db – Caracteres para serem impressos nos centros dos blocos, contém todos os números utilizados no jogo.

Position_config dw – Armazena a posição inicial de impressão para as 16 posições da matriz.

Os demais blocos de memória são referentes a *strings* e seus respectivos comprimentos.

3. Considerações Finais

Uma das principais dificuldades sentidas pelo grupo foi se acostumar com a linguagem de programação Assembly, uma vez que esta requer uma responsabilidade muito maior por parte dos programadores, visto que diversos erros não são informados pelo compilador e só vão gerar problemas durante a execução. Outro desafio foi encontrar a plataforma para desenvolvimento que comportasse as necessidades do trabalho, o Emu8086 apresentava uma ferramenta de depuração muito boa, mas infelizmente não suportava o modo de vídeo, e por esse motivo tivemos que usar o GUI Turbo Assembler o qual não possui modo de debug o que dificulta muito encontrar os erros no programa.

Uma vez superados esses contratempos e acostumados ao paradigma de programação do Assembly, o desenvolvimento do jogo ocorreu de forma tranquila e pode-se dizer até prazerosa.